BLM2041 Signals and Systems

Week 1

The Instructors:

Prof. Dr. Nizamettin Aydın

naydin@yildiz.edu.tr

Asist. Prof. Dr. Ferkan Yilmaz

ferkan@yildiz.edu.tr

Information Systems:

Fundamentals

Informatics

- The term informatics broadly describes the study and practice of
  - creating,
  - storing,
  - finding,
  - manipulating
  - sharing

  information.

Informatics - Etymology

- In 1956 the German computer scientist Karl Steinbuch coined the word Informatik
  - [*Informatik: Automatische Informationsverarbeitung* ("Informatics: Automatic Information Processing")]
- The French term informatique was coined in 1962 by Philippe Dreyfus
  - [Dreyfus, Phillipe. L'informatique. Gestion, Paris, June 1962, pp. 240–41]
- The term was coined as a combination of information and automatic to describe the science of automating information interactions

Informatics - Etymology

- The morphology—informat-ion + -ics—uses
- the accepted form for names of sciences,
  - as conics, linguistics, optics,
- or matters of practice,
  - as economics, politics, tactics
- linguistically, the meaning extends easily
  - to encompass both
    - the science of information
    - the practice of information processing.

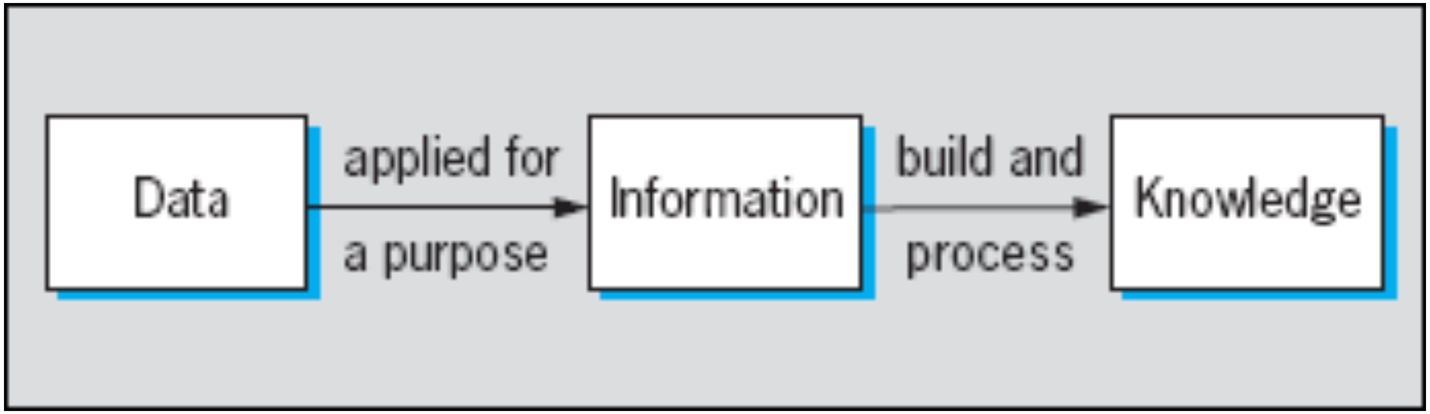
Data - Information - Knowledge

- Data
  - unprocessed facts and figures without any added interpretation or analysis.
    - {The price of crude oil is $80 per barrel.}
- Information
  - data that has been interpreted so that it has meaning for the user.
    - {The price of crude oil has risen from $70 to $80 per barrel}
      - [gives meaning to the data and so is said to be information to someone who tracks oil prices.]

Data - Information - Knowledge

- Knowledge
  - a combination of information, experience and insight that may benefit the individual or the organisation.
    - {When crude oil prices go up by $10 per barrel, it's likely that petrol prices will rise by 2p per litre.}
      - [This is knowledge]
      - [insight: the capacity to gain an accurate and deep understanding of someone or something; an accurate and deep understanding]

Converting data into information



- Data becomes information when it is applied to some purpose and adds value for the recipient.
  - For example a set of raw sales figures is data.
    - For the Sales Manager tasked with solving a problem of poor sales in one region, or deciding the future focus of a sales drive, the raw data needs to be processed into a sales report.
  - It is the sales report that provides information.

Converting data into information

- Collecting data is expensive
  - you need to be very clear about why you need it and how you plan to use it.
  - One of the main reasons that organisations collect data is to monitor and improve performance.
    - if you are to have the information you need for control and performance improvement, you need to:
      - collect data on the indicators that really do affect performance
      - collect data reliably and regularly
      - be able to convert data into the information you need.

Converting data into information

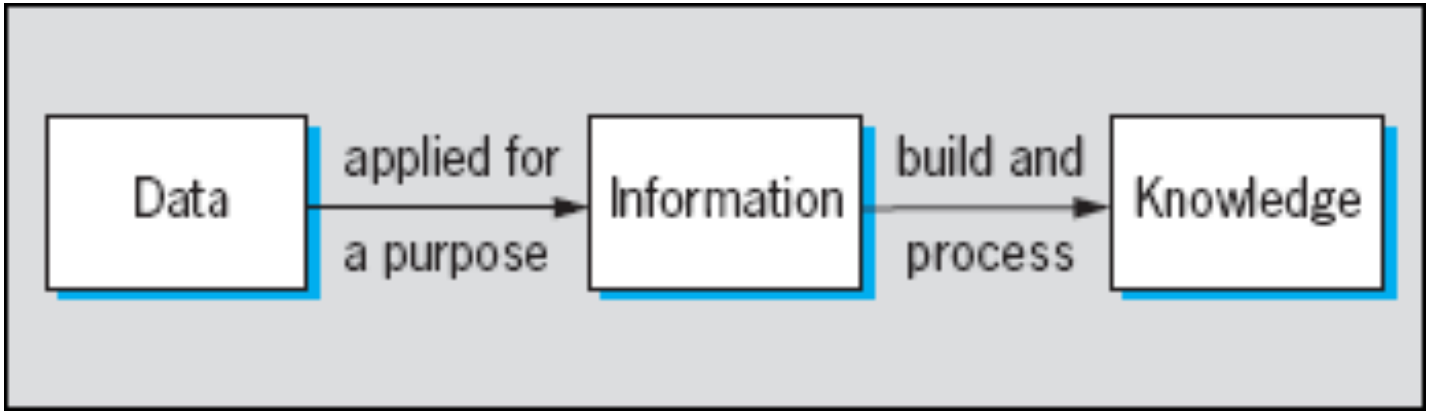
- To be useful, data must satisfy a number of conditions. It must be:
  - relevant to the specific purpose
  - complete
  - accurate
  - timely
    - data that arrives after you have made your decision is of no value

Converting data into information

- in the right format
  - information can only be analysed using a spreadsheet if all the data can be entered into the computer system
- available at a suitable price
  - the benefits of the data must merit the cost of collecting or buying it.

- The same criteria apply to information.
  - It is important
    - to get the right information
    - to get the information right

Converting information to knowledge



- Ultimately the tremendous amount of information that is generated is only useful if it can be applied to create knowledge within the organisation.
- There is considerable blurring and confusion between the terms information and knowledge.

Converting information to knowledge

- think of knowledge as being of two types:
  - Formal, explicit or generally available knowledge.
    - This is knowledge that has been captured and used to develop policies and operating procedures for example.
  - Instinctive, subconscious, tacit or hidden knowledge.
    - Within the organisation there are certain people who hold specific knowledge or have the 'know how'
      - {"I did something very similar to that last year and this happened….."}

Converting information to knowledge

- Clearly, both types of knowledge are essential for the organisation.
- Information on its own will not create a knowledge-based organisation
  - but it is a key building block.
- The right information fuels the development of intellectual capital
  - which in turns drives innovation and performance improvement.

Definition(s) of system

A system can be broadly defined as an integrated set of elements that accomplish a defined objective.

People from different engineering disciplines have different perspectives of what a "system" is.

For example,

- software engineers often refer to an integrated set of computer programs as a "system"
- electrical engineers might refer to complex integrated circuits or an integrated set of electrical units as a "system"

As can be seen, "system" depends on one’s perspective, and the “integrated set of elements that accomplish a defined objective” is an appropriate definition.

Definition(s) of system

- A system is an assembly of parts where:
  - The parts or components are connected together in an organized way.
  - The parts or components are affected by being in the system (and are changed by leaving it).
  - The assembly does something.
  - The assembly has been identified by a person as being of special interest.
- Any arrangement which involves the handling, processing or manipulation of resources of whatever type can be represented as a system.
- Some definitions on online dictionaries
  - http://en.wikipedia.org/wiki/System
  - http://dictionary.reference.com/browse/systems
  - http://www.businessdictionary.com/definition/system.html

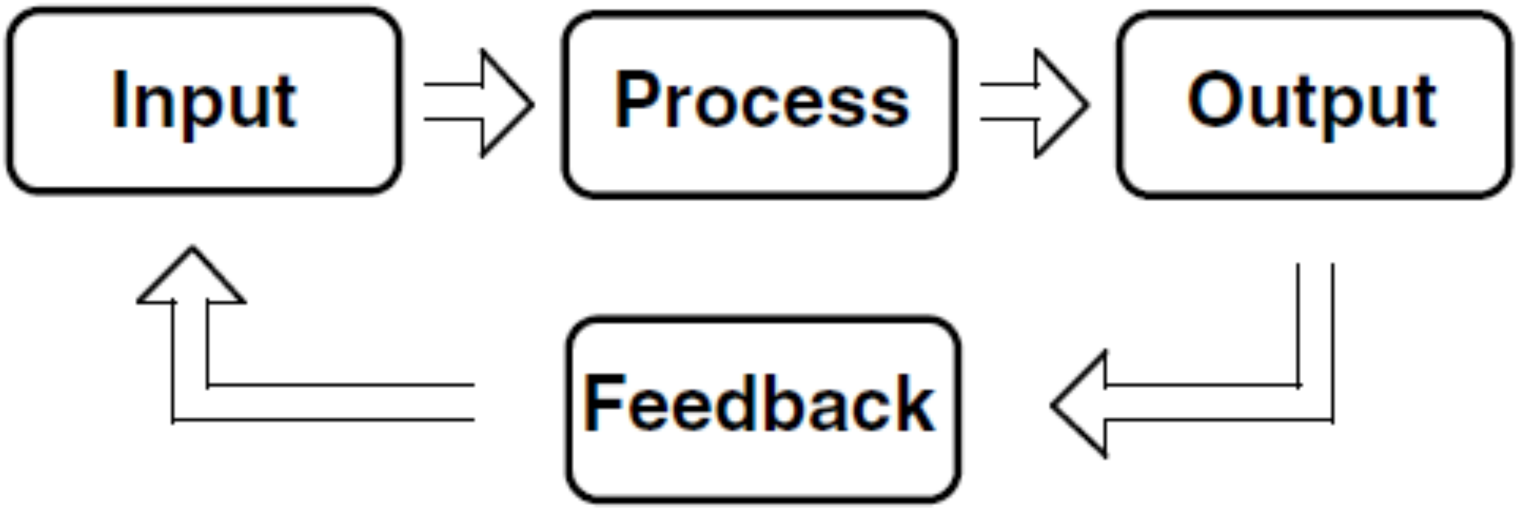
Definition(s) of system

- A system is defined as multiple parts working together for a common purpose or goal.
- Systems can be large and complex
  - such as the air traffic control system or our global telecommunication network.
- Small devices can also be considered as systems
  - such as a pocket calculator, alarm clock, or 10-speed bicycle.

Definition(s) of system

- Systems have inputs, processes, and outputs.
- When feedback (direct or indirect) is involved, that component is also important to the operation of the system.
- To explain all this, systems are usually explained using a model.
- A model helps to illustrate the major elements and their relationship, as illustrated in the next slide

A systems model



Information Systems

- The ways that organizations
  - Store
  - Move
  - Organize
  - Process

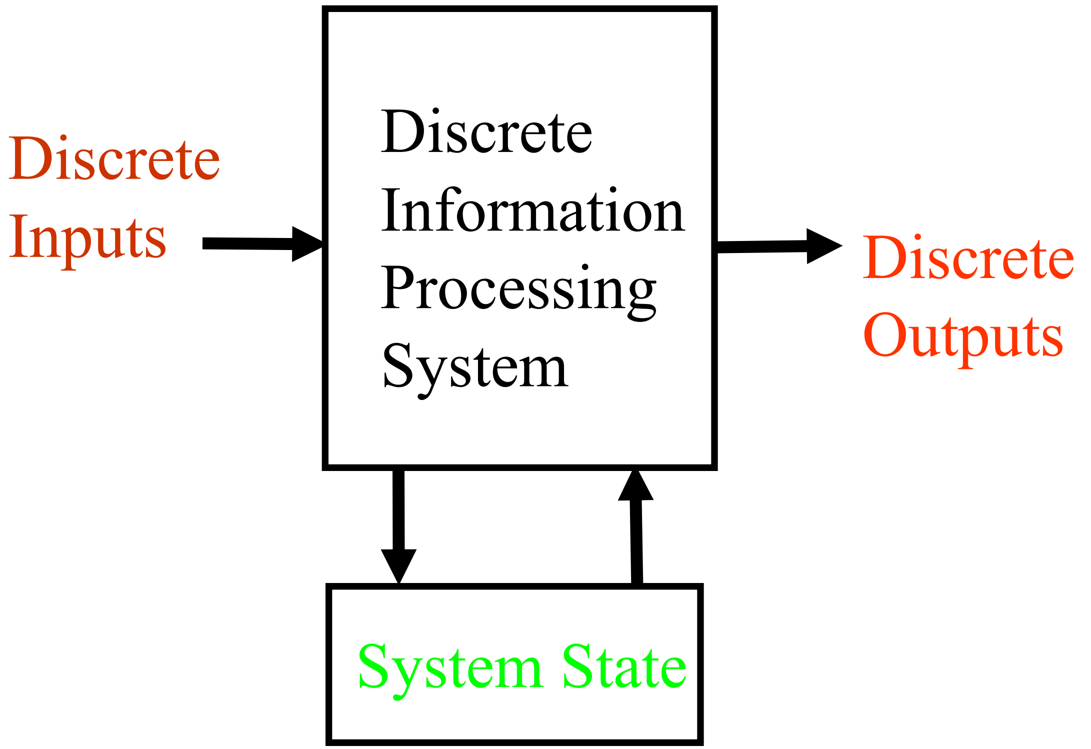
  their information

Information Technology

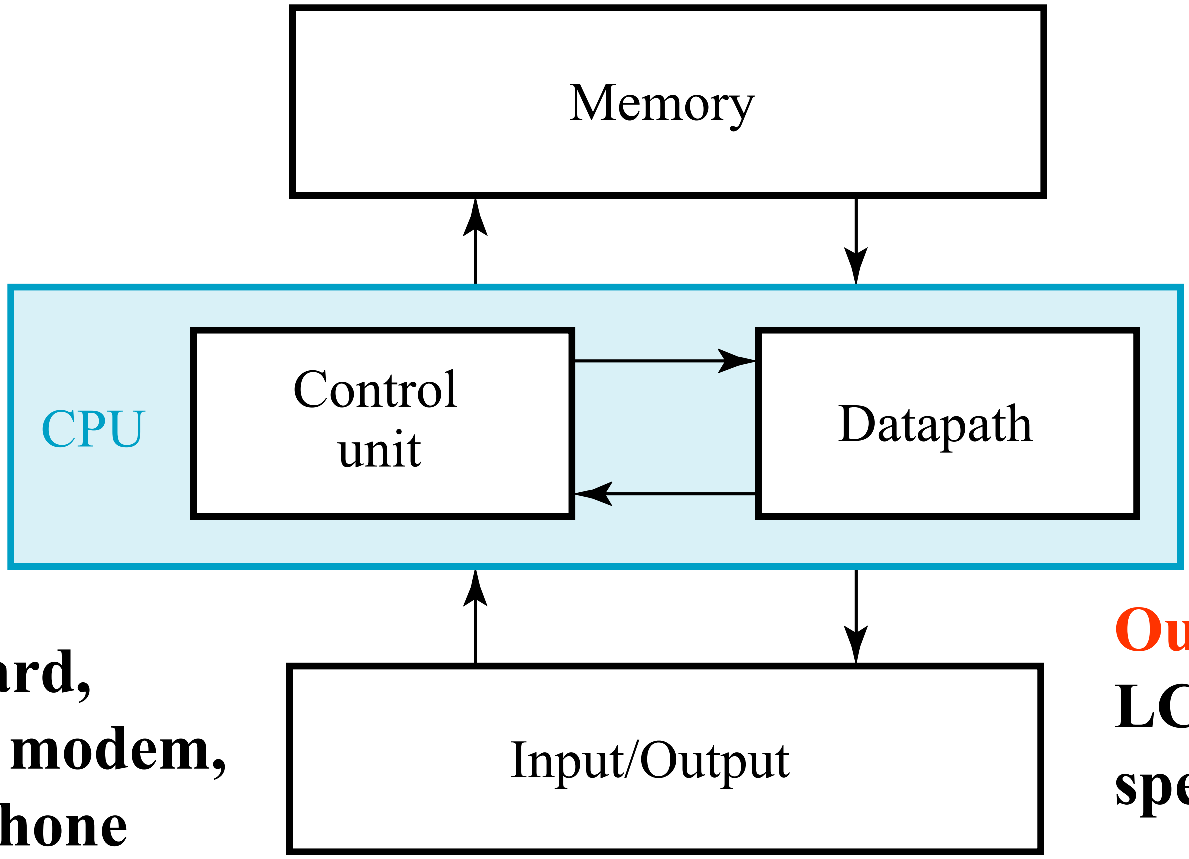
- Components that implement information systems,
  - Hardware
    - physical tools: computer and network hardware, but also low-tech things like pens and paper
  - Software
    - (changeable) instructions for the hardware
  - People
  - Procedures
    - instructions for the people
  - Data/databases

Digital System

- **Takes a set of discrete information (inputs) and discrete internal information (system state) and generates a set of discrete information (outputs).**



A Digital Computer Example



Inputs: Keyboard, mouse, modem, microphone

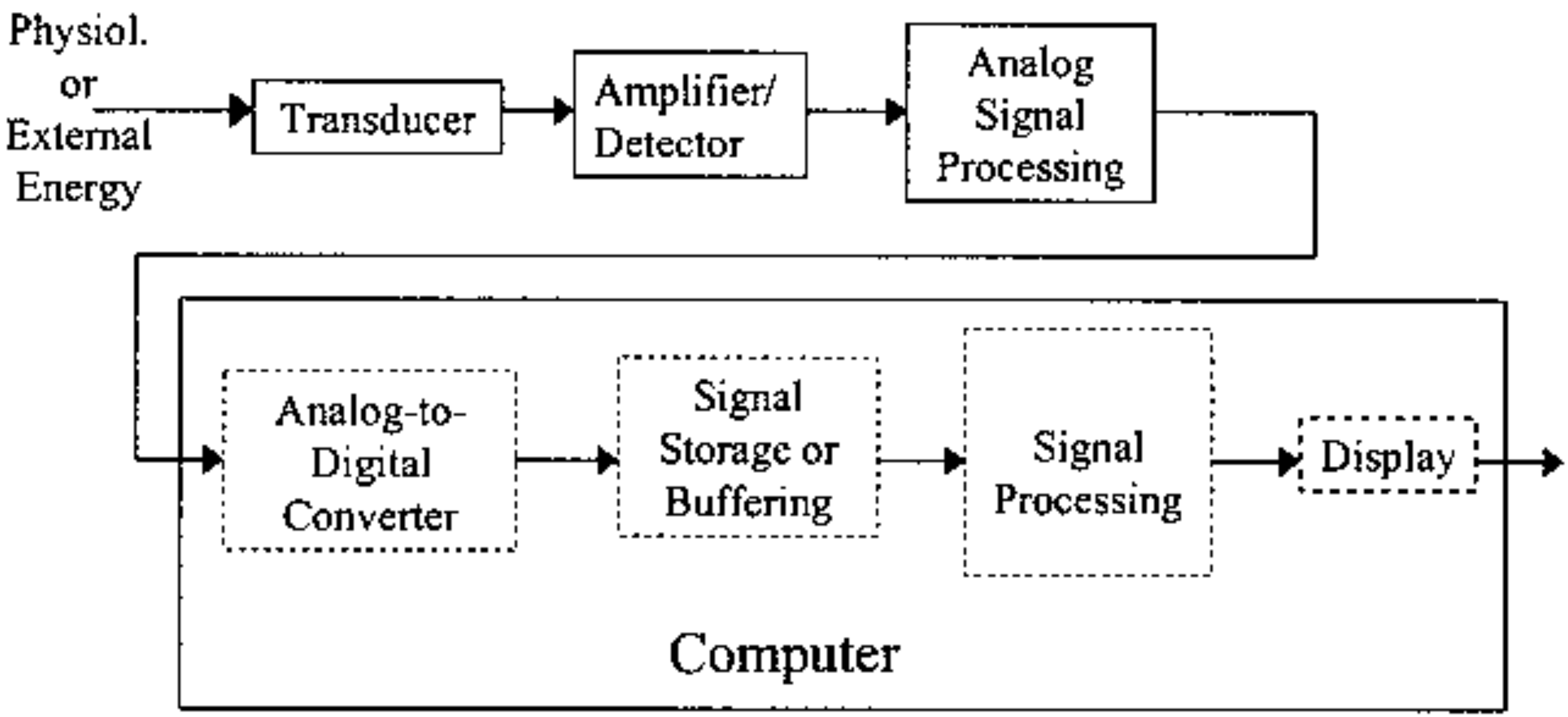
Outputs: CRT, LCD, modem, speakers

Synchronous or Asynchronous?

Signal

- An information variable represented by physical quantity.
- For digital systems, the variable takes on discrete values.
- Two level, or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - words (symbols) Low (L) and High (H)
  - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities

A typical measurement system



Transducers

- A “transducer” is a device that converts energy from one form to another.
- In signal processing applications, the purpose of energy conversion is to transfer information, not to transform energy.
- In physiological measurement systems, transducers may be
  - input transducers (or sensors)
    - they convert a non-electrical energy into an electrical signal.
    - for example, a microphone.
  - output transducers (or actuators)
    - they convert an electrical signal into a non-electrical energy.
    - For example, a speaker.

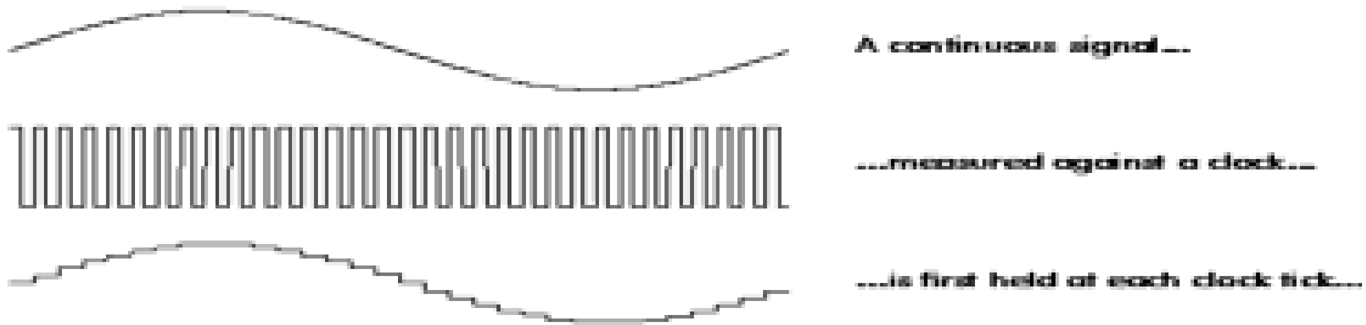
Analogue signal

- The analogue signal
  - a continuous variable defined with infinite precision

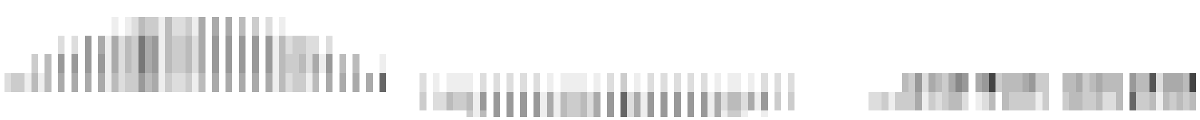
  is converted to a discrete sequence of measured values which are represented digitally
- Information is lost in converting from analogue to digital, due to:
  - inaccuracies in the measurement
  - uncertainty in timing
  - limits on the duration of the measurement
- These effects are called quantisation errors

Digital signal

- The continuous analogue signal has to be held before it can be sampled



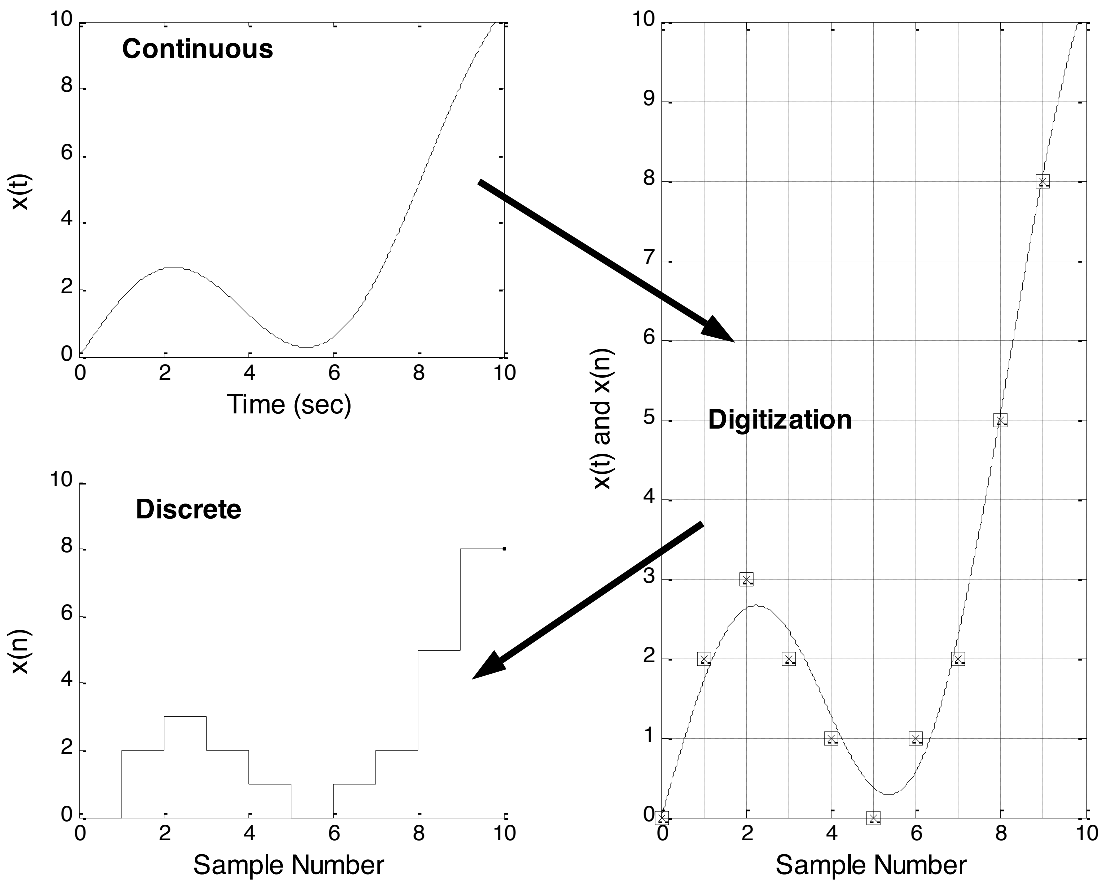
- Otherwise, the signal would be changing during the measurement
- Only after it has been held can the signal be measured, and the measurement converted to a digital value



Signal Encoding: Analog-to Digital Conversion

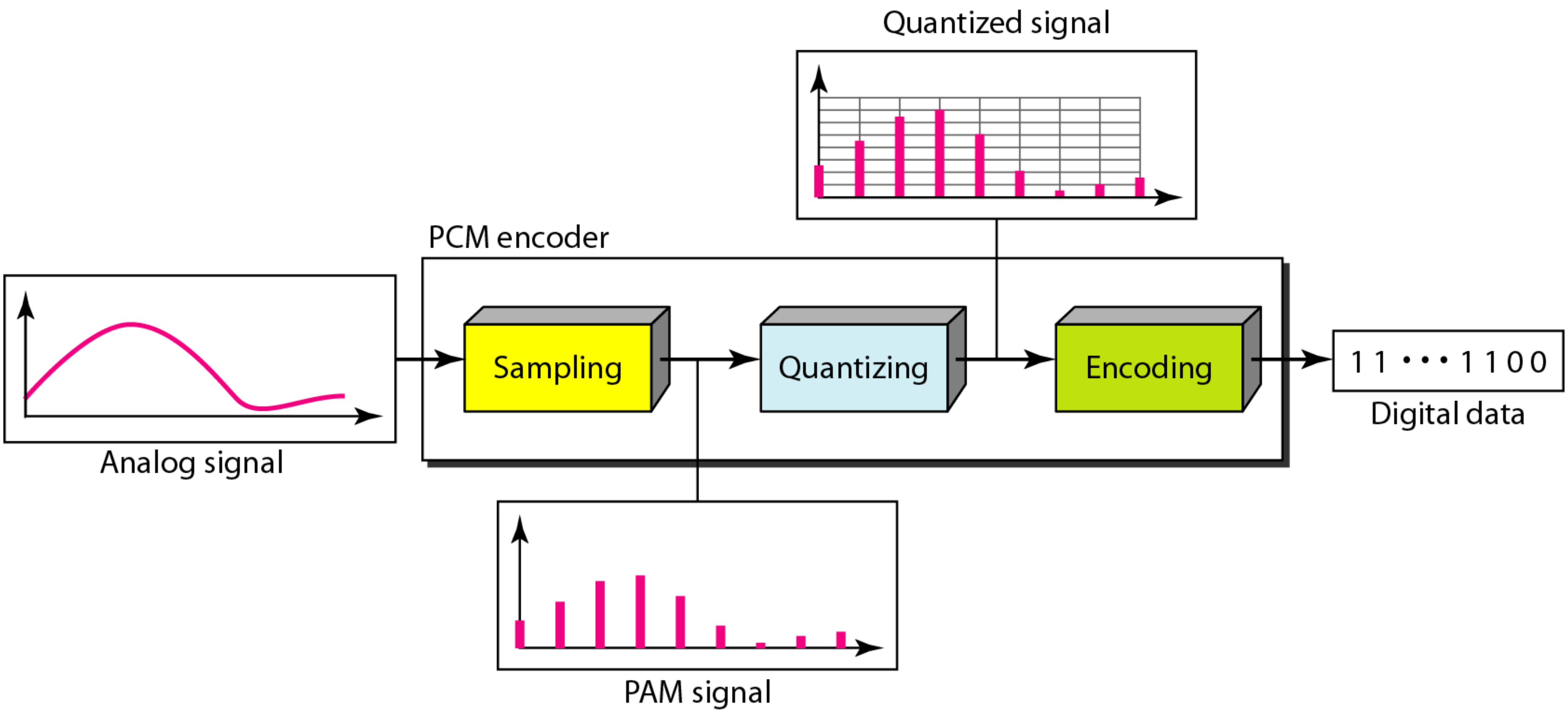
Continuous (analog) signal $\leftrightarrow$ Discrete signal

x(t) = f(t) $\leftrightarrow$ Analog to digital conversion $\leftrightarrow$ x[n] = x [1], x [2], x [3], ... x[n]



Analog-to Digital Conversion

- ADC consists of four steps to digitize an analog signal:
  1. Filtering
  2. Sampling
  3. Quantization
  4. Binary encoding
- Before we sample, we have to filter the signal to limit the maximum frequency of the signal as it affects the sampling rate.
- Filtering should ensure that we do not distort the signal, ie remove high frequency components that affect the signal shape.

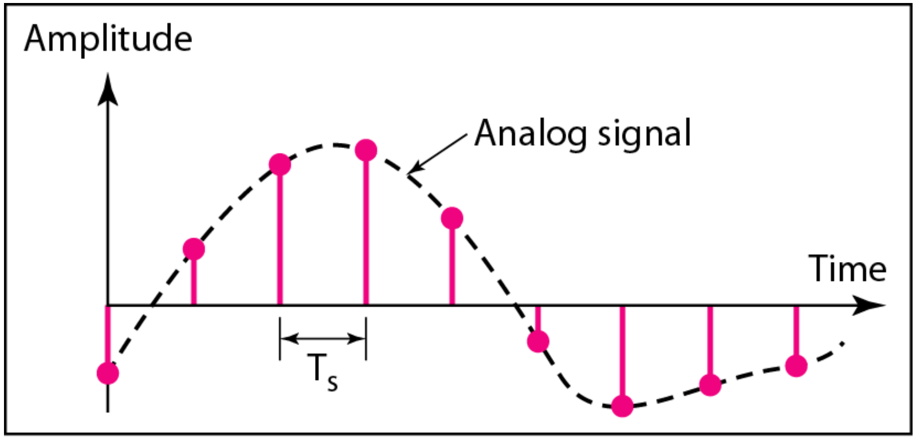

Quantized signal
PCM encoder
Sampling
Quantizing
Encoding
1 1 • • • 1 1 0 0
Digital data
Analog signal
PAM signal

Sampling

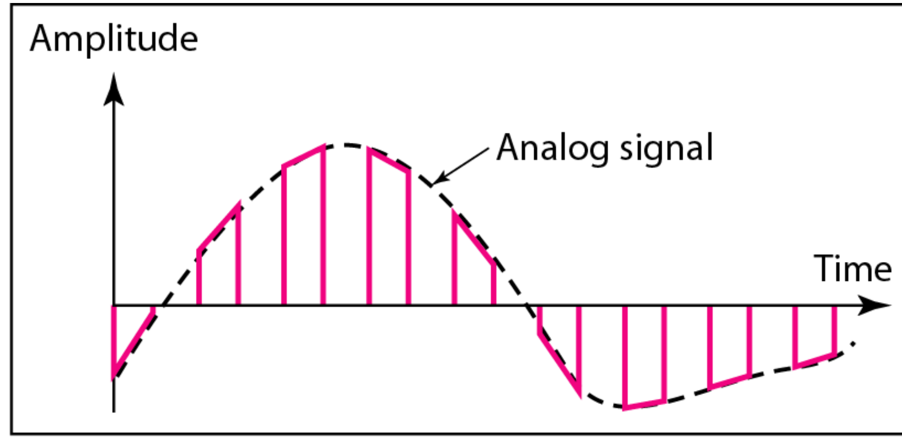
- The sampling results in a discrete set of digital numbers that represent measurements of the signal
  - usually taken at equal intervals of time
- Sampling takes place after the hold
  - The hold circuit must be fast enough that the signal is not changing during the time the circuit is acquiring the signal value
- We don't know what we don't measure
- In the process of measuring the signal, some information is lost

Sampling

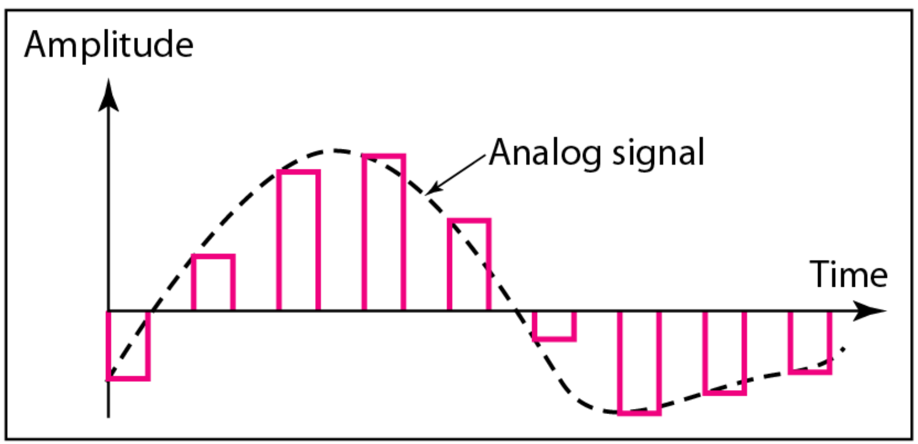
- Analog signal is sampled every T_S secs.
- T_s is referred to as the sampling interval.
- $f_s = 1/T_s$ is called the sampling rate or sampling frequency.
- There are 3 sampling methods:
  - Ideal - an impulse at each sampling instant
  - Natural - a pulse of short width with varying amplitude
  - Flattop - sample and hold, like natural but with single amplitude value
- The process is referred to as pulse amplitude modulation PAM and the outcome is a signal with analog (non integer) values



a. Ideal sampling

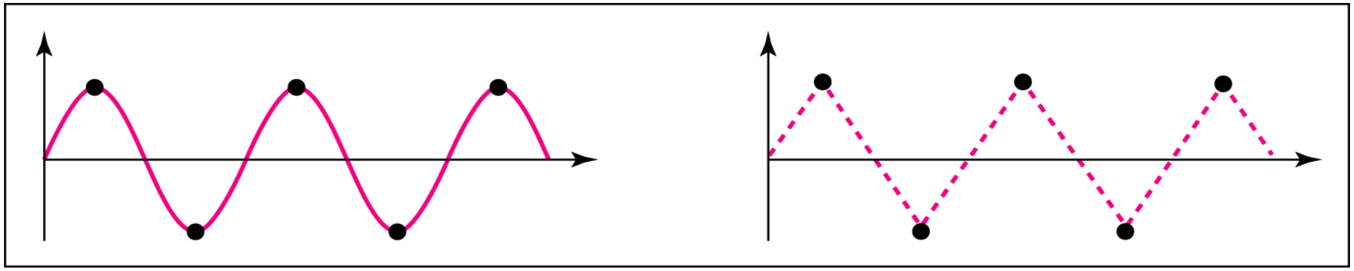


b. Natural sampling

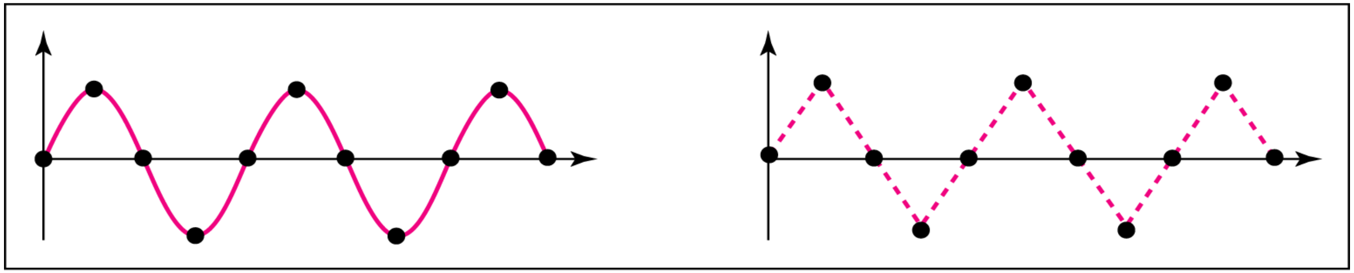


c. Flat-top sampling

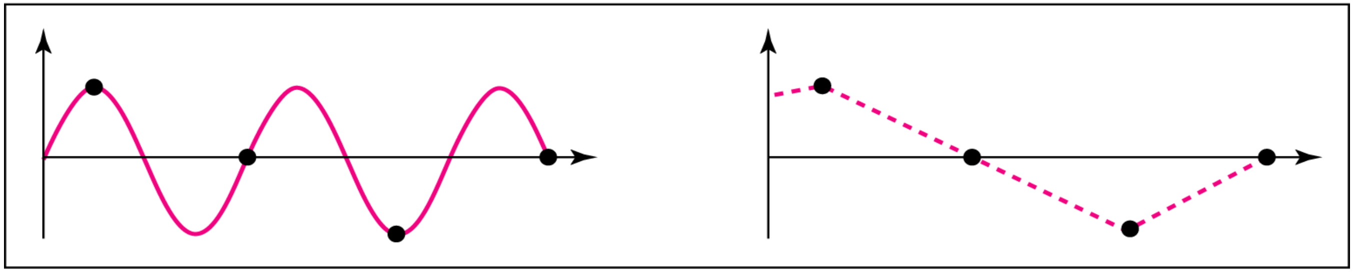
Recovery of a sampled sine wave for different sampling rates



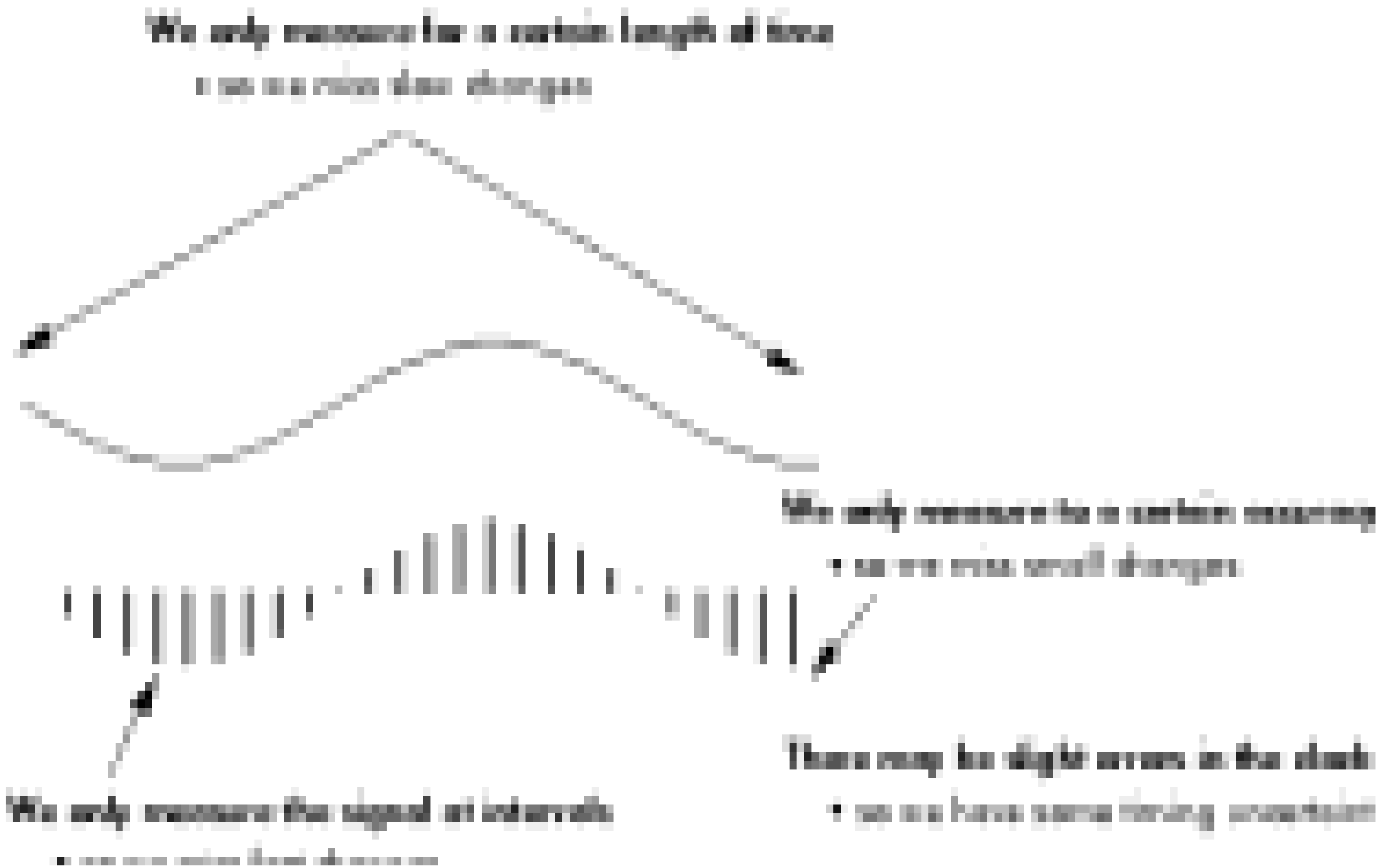
a. Nyquist rate sampling: $f_s = 2\ f$

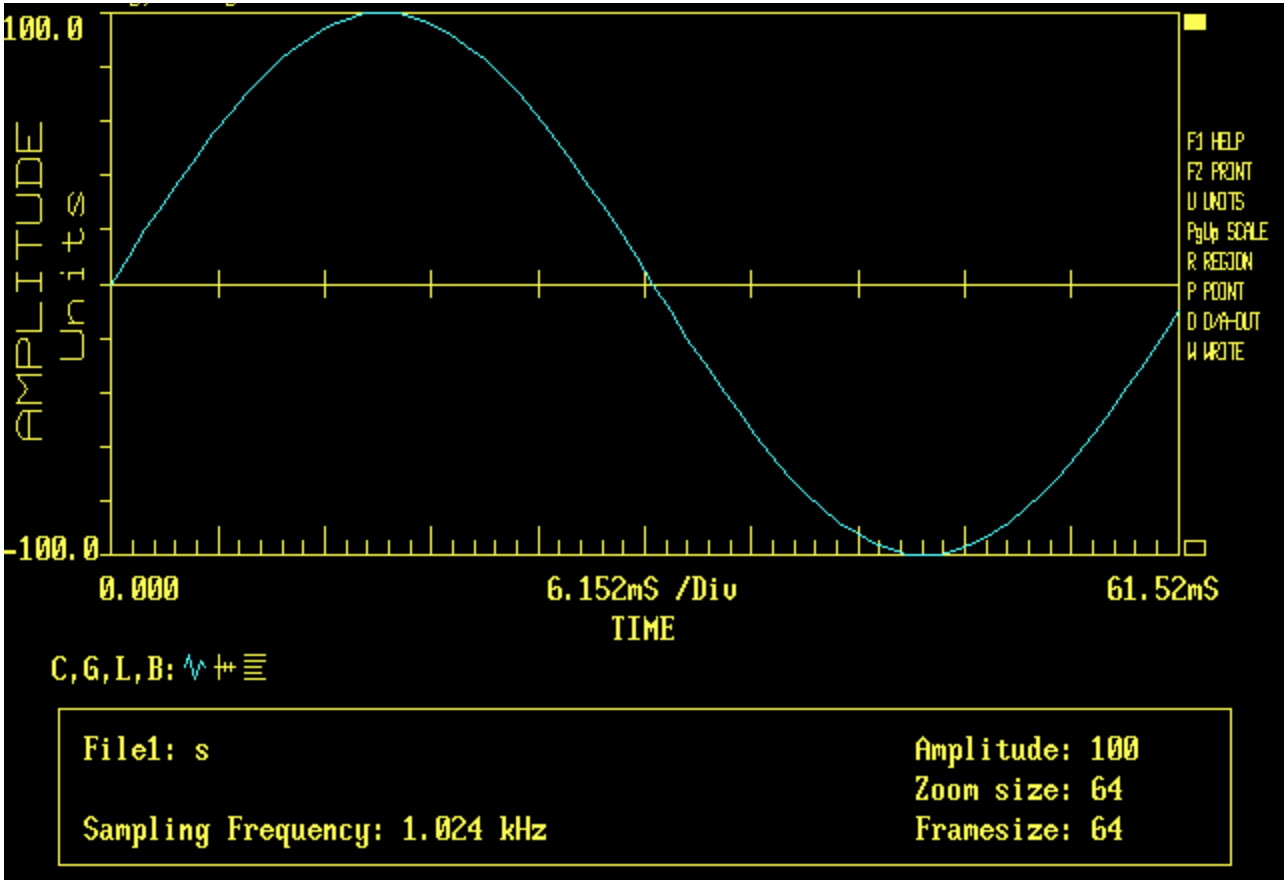


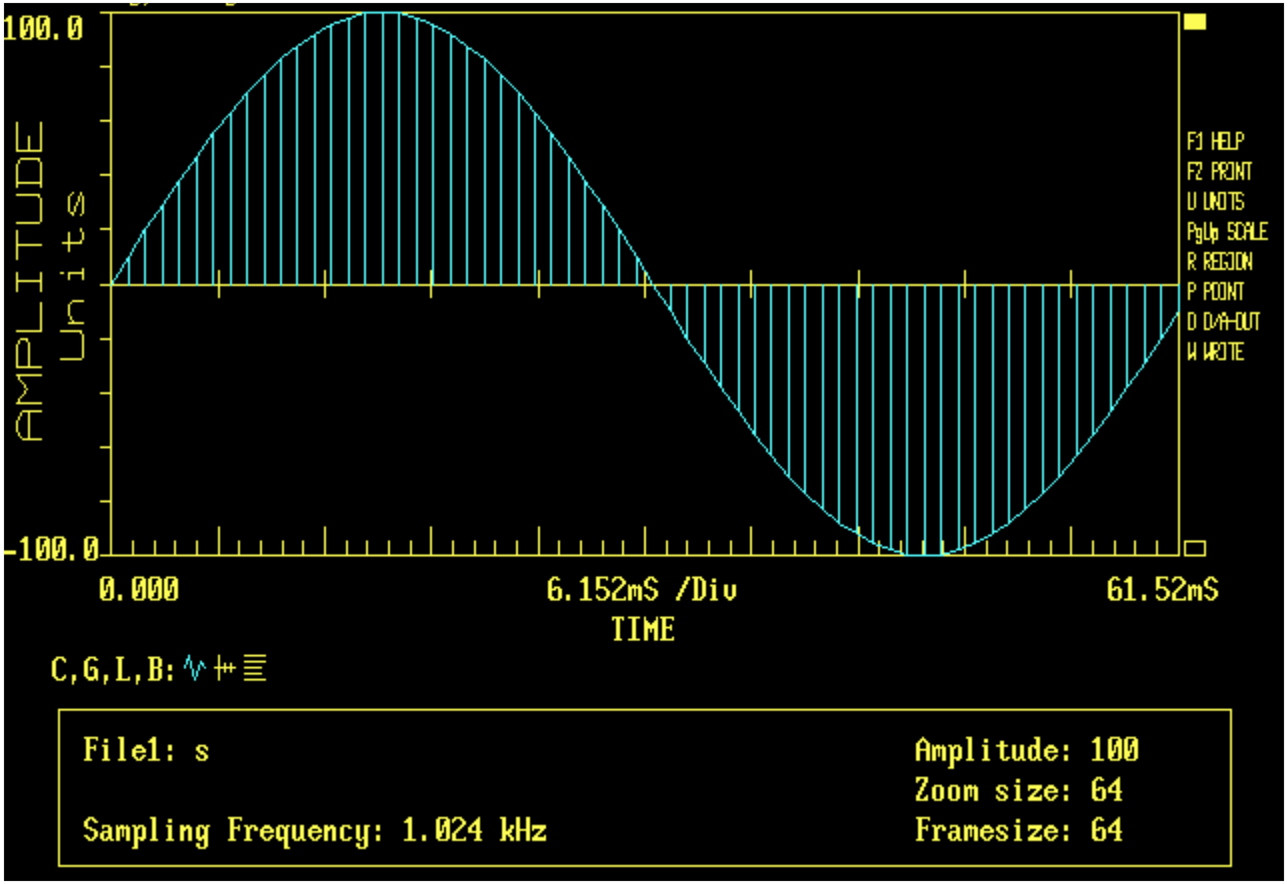
b. Oversampling: $f_s = 4\ f$



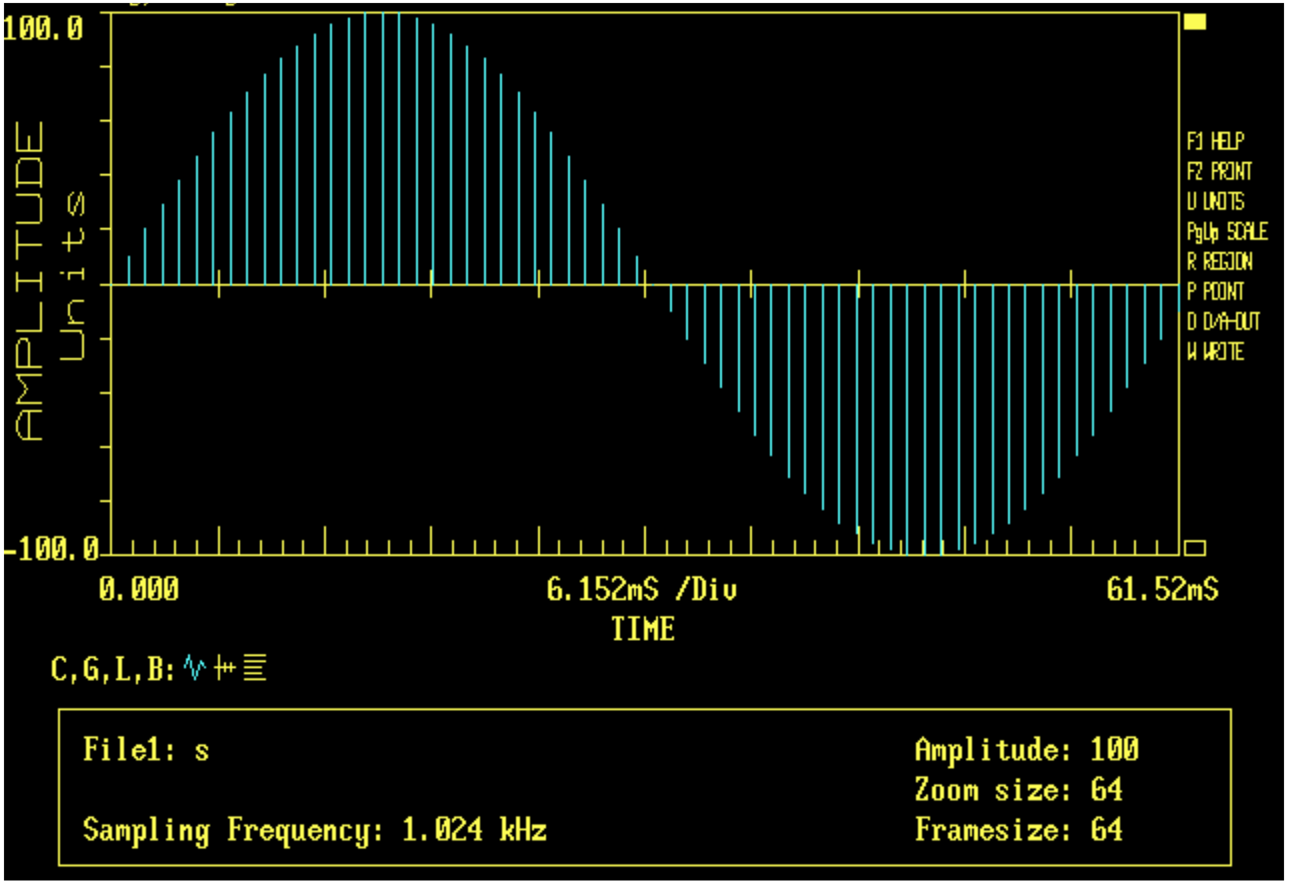
c. Undersampling: $f_s = f$


We only measure for a certain length of time
• so we miss slow changes
We only measure to a certain accuracy
• so we miss small changes
There may be slight errors in the clock
• so we have some timing uncertaint
We only measure the signal at intervals


100.0
-100.0
AMPLITUDE
Units
F1 HELP
F2 PRINT
U UNITS
PgUp SCALE
R REGION
P POINT
D D/A-OUT
W WRITE
0.000
6.152mS /Div
61.52mS
TIME
C,G,L,B:
File1: s
Sampling Frequency: 1.024 kHz
Amplitude: 100
Zoom size: 64
Framesize: 64



100.0
-100.0
AMPLITUDE
Units
F1 HELP
F2 PRINT
U UNITS
PgUp SCALE
R REGION
P POINT
D D/A-OUT
W WRITE
0.000
6.152mS /Div
61.52mS
TIME
C,G,L,B:
File1: s
Sampling Frequency: 1.024 kHz
Amplitude: 100
Zoom size: 64
Framesize: 64

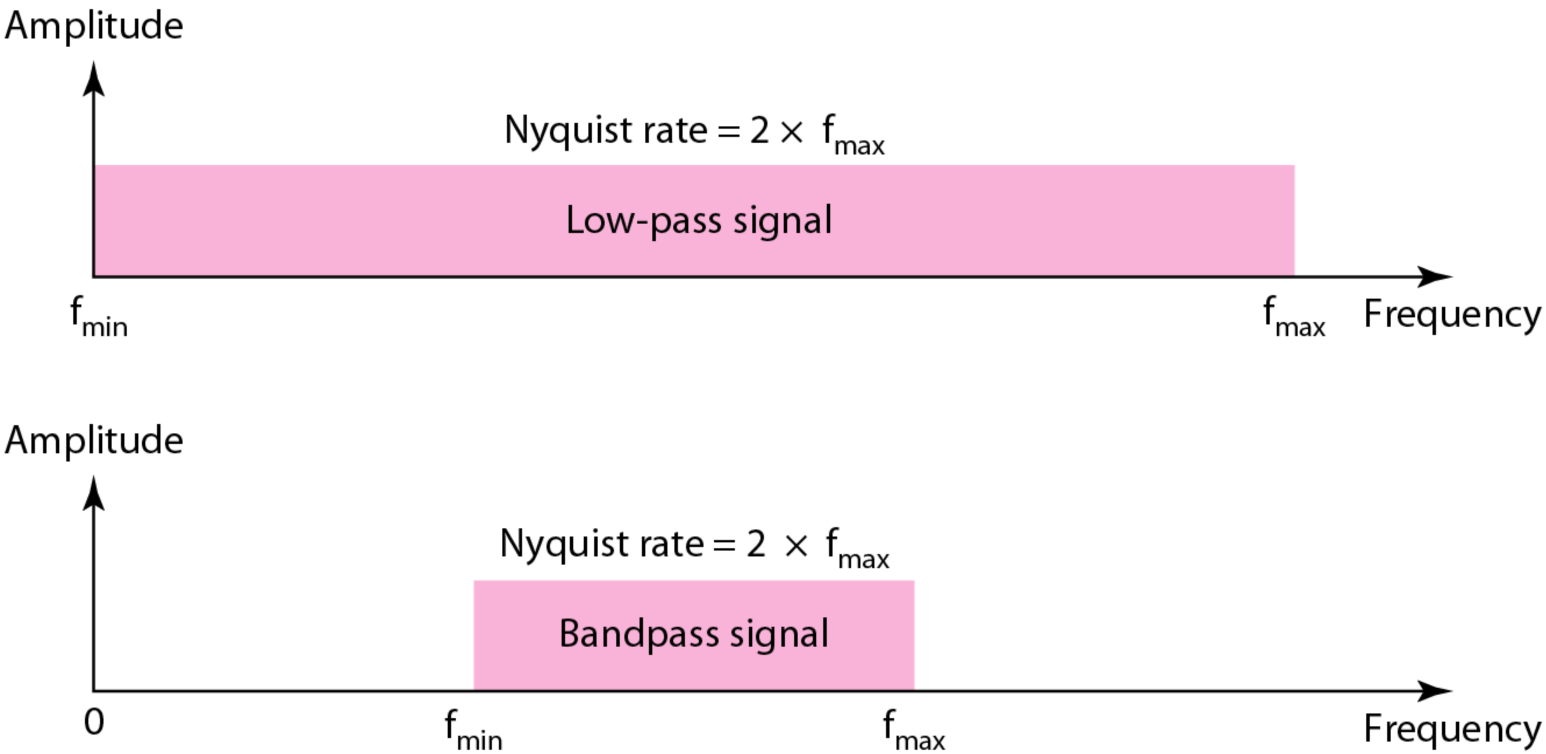

100.0
-100.0
AMPLITUDE
Units
F1 HELP
F2 PRINT
U UNITS
PgUp SCALE
R REGION
P POINT
D D/A-OUT
W WRITE
0.000
6.152mS /Div
61.52mS
TIME
C,G,L,B:
File1: s
Sampling Frequency: 1.024 kHz
Amplitude: 100
Zoom size: 64
Framesize: 64

Sampling Theorem

$$F_s \geq 2f_m$$

According to the Nyquist theorem, the sampling rate must be at least 2 times the highest frequency contained in the signal.

Nyquist sampling rate for low-pass and bandpass signals



Quantization

- Sampling results in a series of pulses of varying amplitude values ranging between two limits: a min and a max.
- The amplitude values are infinite between the two limits.
- We need to map the *infinite* amplitude values onto a finite set of known values.
- This is achieved by dividing the distance between min and max into L zones, each of height Δ.

$$\Delta = (\max - \min)/L$$

Quantization Levels

- The midpoint of each zone is assigned a value from 0 to L-1 (resulting in L values)
- Each sample falling in a zone is then approximated to the value of the midpoint.

Quantization Zones

- Assume we have a voltage signal with amplitutes V_{min}=-20V and V_{max}=+20V.
- We want to use L=8 quantization levels.
- Zone width $\Delta = (20 - -20)/8 = 5$
- The 8 zones are: -20 to -15, -15 to -10, -10 to -5, -5 to 0, 0 to +5, +5 to +10, +10 to +15, +15 to +20
- The midpoints are: -17.5, -12.5, -7.5, -2.5, 2.5, 7.5, 12.5, 17.5

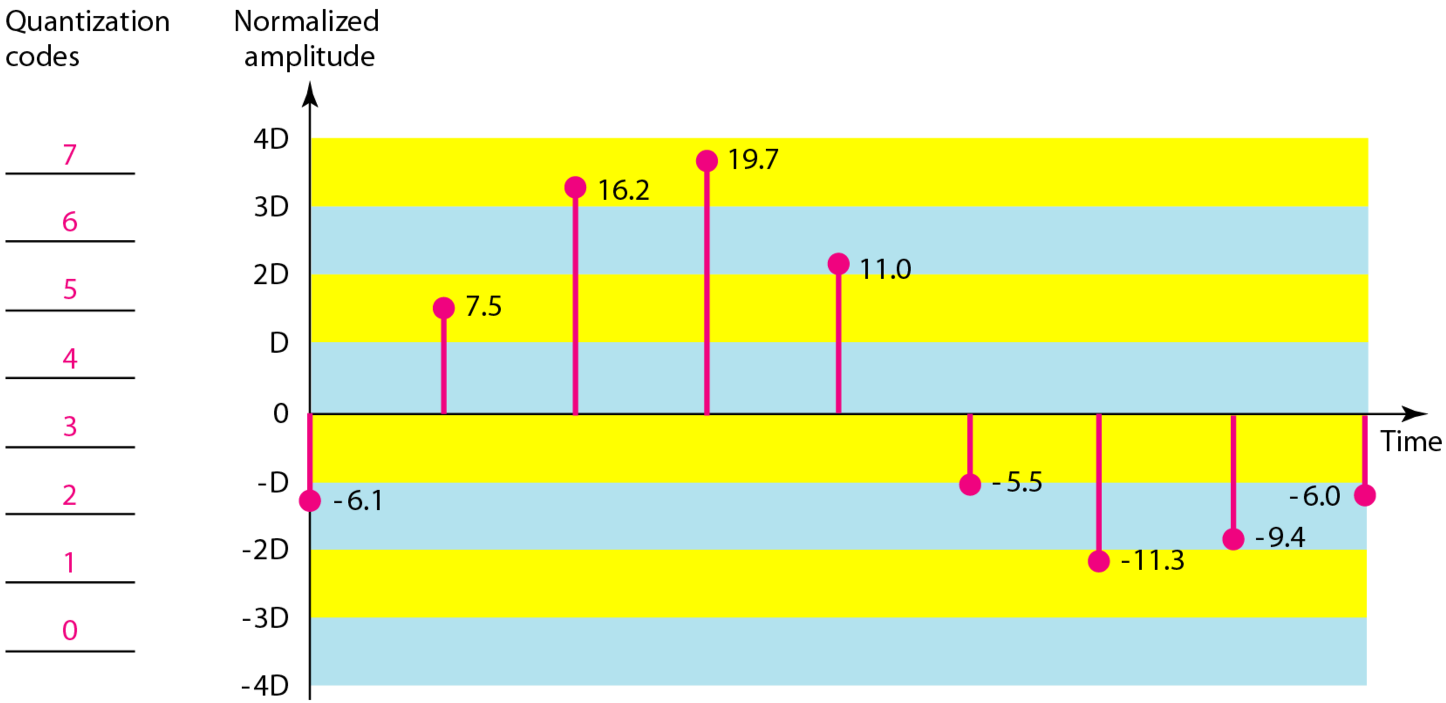
Assigning Codes to Zones

- Each zone is then assigned a binary code.
- The number of bits required to encode the zones, or the number of bits per sample as it is commonly referred to, is obtained as follows:

$$n_b = \log_2 L$$

- Given our example, $n_b = 3$
- The 8 zone (or level) codes are therefore: 000, 001, 010, 011, 100, 101, 110, and 111
- Assigning codes to zones:
  - 000 will refer to zone -20 to -15
  - 001 to zone -15 to -10, etc.

Quantization and encoding of a sampled signal



Normalized PAM values	-1.22	1.50	3.24	3.94	2.20	-1.10	-2.26	-1.88	-1.20
Normalized quantized values	-1.50	1.50	3.50	3.50	2.50	-1.50	-2.50	-1.50	-1.50
Normalized error	-0.38	0	+0.26	-0.44	+0.30	-0.40	-0.24	+0.38	-0.30
Quantization code	2	5	7	7	6	2	1	2	2
Encoded words	010	101	111	111	110	010	001	010	010

Quantization Error

- When a signal is quantized, we introduce an error
  - the coded signal is an approximation of the actual amplitude value.
- The difference between actual and coded value (midpoint) is referred to as the quantization error.
- The more zones, the smaller Δ
  - which results in smaller errors.
- BUT, the more zones the more bits required to encode the samples
  - higher bit rate

Analog-to-digital Conversion

Example An 12-bit analog-to-digital converter (ADC) advertises an accuracy of ± the least significant bit (LSB). If the input range of the ADC is 0 to 10 volts, what is the accuracy of the ADC in analog volts?

<u>Solution:</u>

If the input range is 10 volts then the analog voltage represented by the LSB would be:

$$V_{LSB} = \frac{V_{\max}}{2^{\text{Nu bits}}} = \frac{10}{2^{12}} = \frac{10}{4096} = .0024 \text{ volts}$$

Hence the accuracy would be ± 0.0024 volts.

Sampling related concepts

- Over/exact/under sampling
- Regular/irregular sampling
- Linear/Logarithmic sampling
- Aliasing
- Anti-aliasing filter
- Image
- Anti-image filter

Steps for digitization/reconstruction of a signal

- Band limiting (LPF)
- Sampling / Holding
- Quantization
- Coding

These are basic steps for A/D conversion

- D/A converter
- Sampling / Holding
- Image rejection

These are basic steps for reconstructing a sampled digital signal

Digital data: end product of A/D conversion and related concepts

- Bit: least digital information, binary 1 or 0
- Nibble: 4 bits
- Byte: 8 bits, 2 nibbles
- Word: 16 bits, 2 bytes, 4 nibbles
- Some jargon:
  - integer, signed integer, long integer, 2s complement, hexadecimal, octal, floating point, etc.


MP3 Audio
Recording
ADC
00000
00000
DSP
00000
MEMORY
DAC
00000
00000
Playback

Measures of capacity and speed in Computers

Special Powers of 10 and 2 :

- Kilo- (K) = 1 thousand = 10^3 and 2^{10}
- Mega- (M) = 1 million = 10^6 and 2^{20}
- Giga- (G) = 1 billion = 10^9 and 2^{30}
- Tera- (T) = 1 trillion = 10^{12} and 2^{40}
- Peta- (P) = 1 quadrillion = 10^{15} and 2^{50}

Whether a metric refers to a power of ten or a power of two typically depends upon what is being measured.

Example

- Hertz = clock cycles per second (frequency)
  - 1MHz = 1,000,000Hz
  - Processor speeds are measured in MHz or GHz.
- Byte = a unit of storage
  - 1KB = 2^{10} = 1024 Bytes
  - 1MB = 2^{20} = 1,048,576 Bytes
  - Main memory (RAM) is measured in MB
  - Disk storage is measured in GB for small systems, TB for large systems.

Measures of time and space

- Milli- (m) = 1 thousandth = 10^{-3}
- Micro- (μ) = 1 millionth = 10^{-6}
- Nano- (n) = 1 billionth = 10^{-9}
- Pico- (p) = 1 trillionth = 10^{-12}
- Femto- (f) = 1 quadrillionth = 10^{-15}

Data types

- Our first requirement is to find a way to represent information (data) in a form that is mutually comprehensible by human and machine.
  - Ultimately, we need to develop schemes for representing all conceivable types of information - language, images, actions, etc.
  - Specifically, the devices that make up a computer are switches that can be on or off, i.e. at high or low voltage.
  - Thus they naturally provide us with two symbols to work with:
    - we can call them on and off, or 0 and 1.

What kinds of data do we need to represent?

Numbers

signed, unsigned, integers, floating point, complex, rational, irrational, …

Text

characters, strings, …

Images

pixels, colors, shapes, …

Sound

Logical

true, false

Instructions

…

Data type:

- representation and operations within the computer

Number Systems – Representation

- Positive radix, positional number systems
- A number with *radix* ***r*** is represented by a string of digits:

$$A_{n-1}A_{n-2}\ \cdots\ A_1A_0\ .\ A_{-1}A_{-2}\ \cdots\ A_{-m+1}A_{-m}$$

in which $\mathbf{0} \le A_i < r$ and **.** is the *radix point*.

- The string of digits represents the power series:

$$(\text{Number})_r = \left(\sum_{i=0}^{i=n-1} A_i \cdot r^i\right) + \left(\sum_{j=-m}^{j=-1} A_j \cdot r^j\right)$$

(Integer Portion) + (Fraction Portion)

Decimal Numbers

- "decimal" means that we have ten digits to use in our representation
  - the symbols 0 through 9
- What is 3546?
  - it is *three* thousands plus *five* hundreds plus *four* tens plus *six* ones.
  - i.e. $3546 = 3 \times 10^3 + 5 \times 10^2 + 4 \times 10^1 + 6 \times 10^0$
- How about negative numbers?
  - we use two more symbols to distinguish positive and negative:

    \+ and -

Decimal Numbers

- "decimal" means that we have <u>ten</u> digits to use in our representation (the <u>symbols</u> 0 through 9)
- What is 3546?
  - it is *three* <u>thousands</u> plus *five* <u>hundreds</u> plus *four* <u>tens</u> plus *six* <u>ones</u>.
  - i.e. $3546 = 3.10^3 + 5.10^2 + 4.10^1 + 6.10^0$
- How about negative numbers?
  - we use two more <u>symbols</u> to distinguish positive and negative:

    + and **-**

Unsigned Binary Integers

$Y = \text{“abc”} = a.2^2 + b.2^1 + c.2^0$

(where the digits a, b, c can each take on the values of 0 or 1 only)

N = number of bits

Range is:
$0 \leq i < 2^N - 1$

	3-bits	5-bits	8-bits
0	000	00000	00000000
1	001	00001	00000001
2	010	00010	00000010
3	011	00011	00000011
4	100	00100	00000100

Problem:

- How do we represent *negative* numbers?

Signed Binary Integers
-2s Complement representation-

- Transformation
  - To transform a into -a, invert all bits in a and add 1 to the result

Range is:
$-2^{N-1} < i < 2^{N-1} - 1$

Advantages:

- Operations need not check the sign
- Only one representation for zero
- Efficient use of all the bits

-16	10000
…	…
-3	11101
-2	11110
-1	11111
0	00000
+1	00001
+2	00010
+3	00011
…	…
+15	01111

Limitations of integer representations

- Most numbers are not integer!
  - Even with integers, there are two other considerations:
- Range:
  - The magnitude of the numbers we can represent is determined by how many bits we use:
    - e.g. with 32 bits the largest number we can represent is about +/- 2 billion, far too small for many purposes.
- Precision:
  - The exactness with which we can specify a number:
    - e.g. a 32 bit number gives us 31 bits of precision, or roughly 9 figure precision in decimal repesentation.
- We need another data type!

Real numbers

- Our decimal system handles non-integer *real* numbers by adding yet another symbol - the decimal point (**.**) to make a *fixed point* notation:
  - e.g. $3456.78 = 3.10^3 + 4.10^2 + 5.10^1 + 6.10^0 + 7.10^{-1} + 8.10^{-2}$
- The *floating point*, or scientific, notation allows us to represent very large and very small numbers (integer or real), with as much or as little precision as needed:
  - Unit of electric charge $e = 1.602\ 176\ 462 \times 10^{-19}$ Coulomb
  - Volume of universe = 1×10^{85} cm^3
    - the two components of these numbers are called the mantissa and the exponent

Real numbers in binary

- We mimic the decimal floating point notation to create a "hybrid" binary floating point number:
  - We first use a "binary point" to separate whole numbers from fractional numbers to make a fixed point notation:
    - e.g. 00011001.110 = $1.2^4 + 1.10^3 + 1.10^1 + 1.2^{-1} + 1.2^{-2}$ => 25.75

      ($2^{-1} = 0.5$ and $2^{-2} = 0.25$, etc.)
  - We then "float" the binary point:
    - 00011001.110 => 1.1001110×2^4

      mantissa = 1.1001110, exponent = 4
  - Now we have to express this without the extra symbols (x, 2, .)
    - by convention, we divide the available bits into three fields:

sign, mantissa, exponent

IEEE-754 fp numbers - 1

	s	biased exp.	fraction
32 bits:	1	8 bits	23 bits

$$N = (-1)^{s} \times 1.\text{fraction} \times 2^{(\text{biased exp.} - 127)}$$

- Sign: 1 bit
- Mantissa: 23 bits
  - We "normalize" the mantissa by dropping the leading 1 and recording only its fractional part (why?)
- Exponent: 8 bits
  - In order to handle both +ve and -ve exponents, we add 127 to the actual exponent to create a "biased exponent":
    - 2^{-127} => biased exponent = 0000 0000 (= 0)
    - 2^{0} => biased exponent = 0111 1111 (= 127)
    - 2^{+127} => biased exponent = 1111 1110 (= 254)

IEEE-754 fp numbers - 2

- Example: Find the corresponding fp representation of 25.75
    - 25.75 => 00011001.110 => 1.1001110 x 2^4
    - sign bit = 0 (+ve)
    - normalized mantissa (fraction) = 100 1110 0000 0000 0000 0000
    - biased exponent = 4 + 127 = 131 => 1000 0011
    - so 25.75 => 0 1000 0011 100 1110 0000 0000 0000 0000 => x41CE0000
- Values represented by convention:
    - Infinity (+ and -): exponent = 255 (1111 1111) and fraction = 0
    - NaN (not a number): exponent = 255 and fraction $\neq$ 0
    - Zero (0): exponent = 0 and fraction = 0
        - note: exponent = 0 => fraction is *de-normalized,* i.e no hidden 1

IEEE-754 fp numbers - 3

- Double precision (64 bit) floating point

	s	biased exp.	fraction
64 bits:	1	11 bits	52 bits

$$N = (-1)^{s} \times 1.\text{fraction} \times 2^{(\text{biased exp.} - 1023)}$$

- Range & Precision:
  - 32 bit:
    - mantissa of 23 bits + 1 => approx. 7 digits decimal
    - $2^{+/-127}$ => approx. $10^{+/-38}$
  - 64 bit:
    - mantissa of 52 bits + 1 => approx. 15 digits decimal
    - $2^{+/-1023}$ => approx. $10^{+/-306}$

Binary Numbers and Binary Coding

- Flexibility of representation
  - Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.
- Information Types
  - Numeric
    - Must represent range of data needed
    - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
    - Tight relation to binary numbers
  - Non-numeric
    - Greater flexibility since arithmetic operations not applied.
    - Not tied to binary numbers

Non-numeric Binary Codes

- Given n binary digits (called <u>bits</u>), a <u>binary code</u> is a mapping from a set of <u>represented elements</u> to a subset of the 2^n binary numbers.
- Example: A binary code for the seven colors of the rainbow
- Code 100 is not used

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Number of Bits Required

- Given M elements to be represented by a binary code, the minimum number of bits, n, needed, satisfies the following relationships:

  $2^n > M > 2^{(n-1)}$

  $n = \lceil \log_2 M \rceil$ where $\lceil x \rceil$, called the *ceiling function,* is the integer greater than or equal to x.

- Example: How many bits are required to represent decimal digits with a binary code?
  - 4 bits are required ($n = \lceil \log_2 9 \rceil = 4$)

Number of Elements Represented

- Given n digits in radix r, there are r^n distinct elements that can be represented.
- But, you can represent m elements, $m < r^n$
- Examples:
  - You can represent 4 elements in radix $r = 2$ with $n = 2$ digits: (00, 01, 10, 11).
  - You can represent 4 elements in radix $r = 2$ with $n = 4$ digits: (0001, 0010, 0100, 1000).

Binary Coded Decimal (BCD)

- In the 8421 Binary Coded Decimal (BCD) representation each decimal digit is converted to its 4-bit pure binary equivalent
- This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number,
  - but only encodes the first ten values from 0 to 9.
    - For example: $(57)_{dec}$ ➔ $(?)_{bcd}$

(5 7) dec

= (0101 0111)bcd

Error-Detection Codes

- Redundancy (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
- A simple form of redundancy is parity, an extra bit appended onto the code word to make the number of 1's odd or even.
  - Parity can detect all single-bit errors and some multiple-bit errors.
- A code word has even parity if the number of 1's in the code word is even.
- A code word has odd parity if the number of 1's in the code word is odd.

4-Bit Parity Code Example

- Fill in the even and odd parity bits:

Even Parity Message - Parity	Odd Parity Message - Parity
000 _	000 _
001 _	001 _
010 _	010 _
011 _	011 _
100 _	100 _
101 _	101 _
110 _	110 _
111 _	111 _

- The codeword "1111" has <u>even parity</u> and the codeword "1110" has <u>odd parity</u>. Both can be used to represent 3-bit data.

ASCII Character Codes

- American Standard Code for Information Interchange
- This code is a popular code used to represent information sent as character-based data.
- It uses 7- bits to represent
  - 94 Graphic printing characters
  - 34 Non-printing characters
- Some non-printing characters are used for text format
  - e.g. BS = Backspace, CR = carriage return
- Other non-printing characters are used for record marking and flow control
  - e.g. STX = start text areas, ETX = end text areas.

ASCII Properties

- **ASCII has some interesting properties:**
- Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}
- Upper case A-Z span 41_{16} to $5A_{16}$
- Lower case a-z span 61_{16} to $7A_{16}$
  – Lower to upper case translation (and vice versa) occurs by flipping bit 6
- Delete (DEL) is all bits set,
  – a carryover from when punched paper tape was used to store messages

UNICODE

- UNICODE extends ASCII to 65,536 universal characters codes
  - For encoding characters in world languages
  - Available in many modern applications
  - 2 byte (16-bit) code words

Warning: Conversion or Coding?

- Do NOT mix up "conversion of a decimal number to a binary number" with "coding a decimal number with a binary code".
- $13_{10} = 1101_2$
  - This is conversion
- $13 \Leftrightarrow 0001\ 0011_{BCD}$
  - This is coding

Another use for bits: Logic

- Beyond numbers
  - *logical variables* can be *true* or *false*, *on* or *off*, etc., and so are readily represented by the binary system.
  - A logical variable A can take the values *false = 0* or *true = 1* only.
  - The manipulation of logical variables is known as Boolean Algebra, and has its own set of operations
    - which are not to be confused with the arithmetical operations.
  - Some basic operations: NOT, AND, OR, XOR

Basic Logic Operations

- Truth Tables of Basic Operations

NOT

A	A'
0	1
1	0

AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

- Equivalent Notations
  - not A = A' = $\overline{A}$
  - A and B = A.B = A∧B = A intersection B
  - A or B = A+B = A∨B = A union B

More Logic Operations

XOR		
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

XNOR		
A	B	$(A \oplus B)'$
0	0	1
0	1	0
1	0	0
1	1	1

– Exclusive OR (XOR): either A or B is 1, not both

– $A \oplus B = A.B' + A'.B$